Amazon SageMaker

Andrew K Jang
Naga Venkata Sai Rama Krishna Rohan Avireddy
Shrey Srivastava

Outline

- Introduction to Amazon SageMaker
- Core Features and Advantages
- Built-in Algorithms and Supported Endpoints
- Distributed Training Deep Dive
- Use Cases and Cost Structure
- Advanced Metrics and Optimization Tips
- Real-Time Model Monitoring
- Hybrid Deployment Strategies
- Integrating SageMaker with MLOps
- Case Study: Accelerating NLP with SageMaker
- Expanding SageMaker Applications
- Live Demo Preview
- Challenges and Future Directions
- Key Takeaways and Q&A

Overview

What is Amazon SageMaker?

- A suite of tools and services provided by Amazon in assisting with the creation, training, and deployment of machine learning models using AWS’s cloud resources.
- Machine learning requires many moving parts
- Sagemaker aims to provide a collaborative end-to-end machine learning platform.

The Machine Learning Pipeline

Data Preparation

- Data Cleaning
- Data Aggregation
- Data Analysis and Transformation
- Data Validation
- Feature Engineering
- Data Splitting

Model Creation

- Model Building
- Model Training
- Model Validation
- Scaled Training

Rollout

- Deployment
- Serving
- Monitoring, Logging, Explainability
- Visualization

History of SageMaker

2019: SageMaker Studio

2020: SageMaker Pipelines, SageMaker Distributed, SageMaker Data Wrangler, SageMaker Model Registry

2021: SageMaker Canvas, SageMaker Endpoints

2022: SageMaker Collaboration, Geospatial Capabilities

2023: New SageMaker Experience, Code Editor

Launch of SageMaker - 2017

Data Preparation	Model Creation	Rollout
	TensorFlow BlazingText DeepAR	Deployment Scaled Deployment CloudTrail

SageMaker Neo, SageMaker Marketplace - 2018

Data Preparation	Model Creation	Rollout
SageMaker Ground Truth	SageMaker Marketplace - Sharing of a wide variety of models for use on AWS infrastructure Semantic Segmentation Reinforcement Learning	Deployment Scaled Deployment CloudTrail Inference Pipelines

SageMaker Studio - 2019

Data Preparation → Model Creation → Rollout

SageMaker Ground Truth

Main Feature

Some Features of SageMaker

- Built-in Algorithms: Includes XGBoost, Linear Learner, and K-Means for diverse ML tasks.
- Training, Tuning & Deployment: Automated tuning and scalable endpoints for real-time, batch, and asynchronous inference.
- Notebook Instances: Managed Jupyter notebooks for development.
- SageMaker Pipelines: Automates end-to-end MLOps workflows.
- Integrations & Framework Support: Works with AWS services (S3, Lambda, Athena) and frameworks like PyTorch, TensorFlow, and Scikit-Learn.

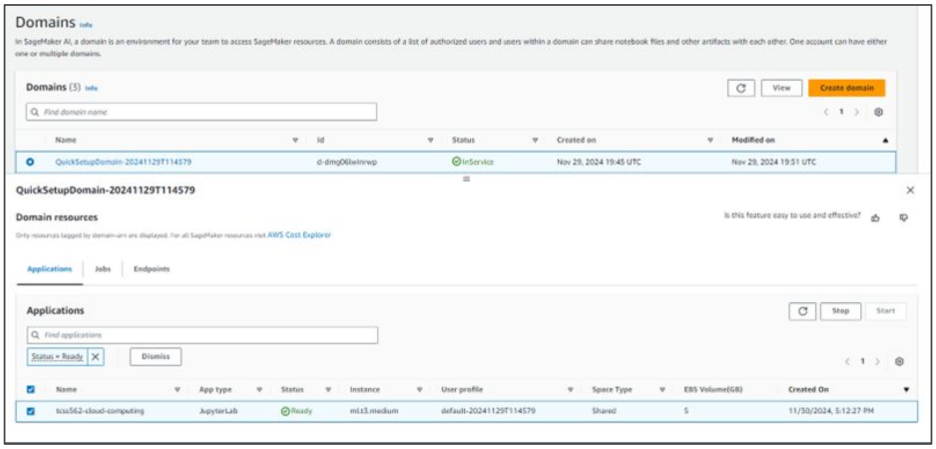
JupyterLab Integration

What is JupyterLab?

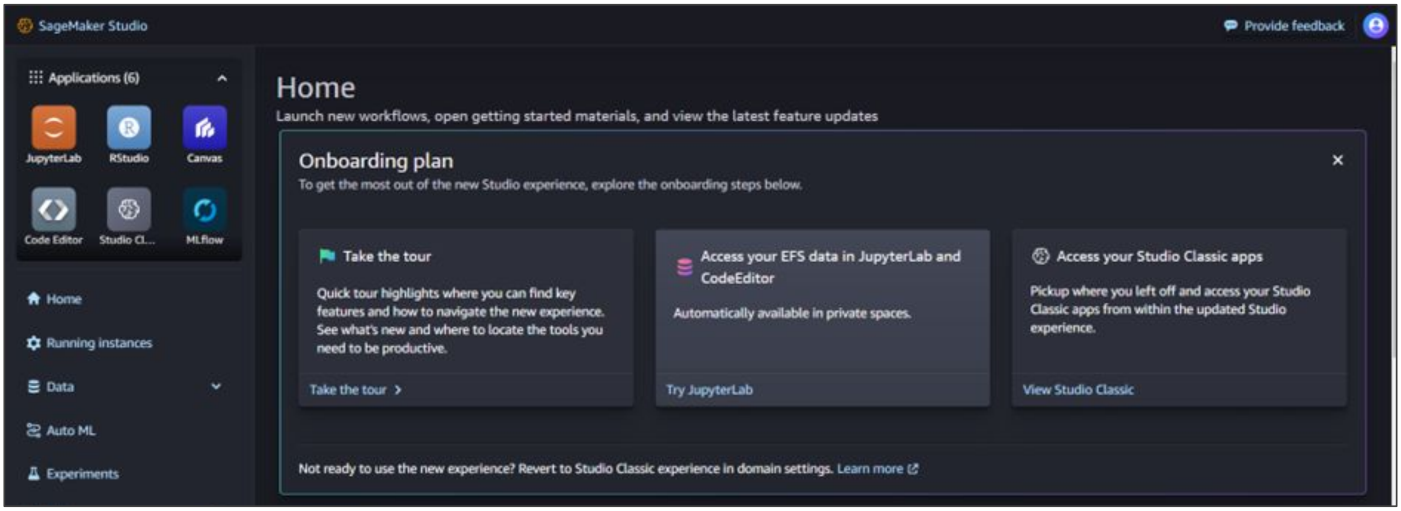
- Interactive development environment
- Launched with a diverse array of instances
- Run multiple individual Jupyter notebooks
- Develop and test scripts for data processing or machine learning.
- With SageMaker, provides a shared collaborative workspace on the cloud.

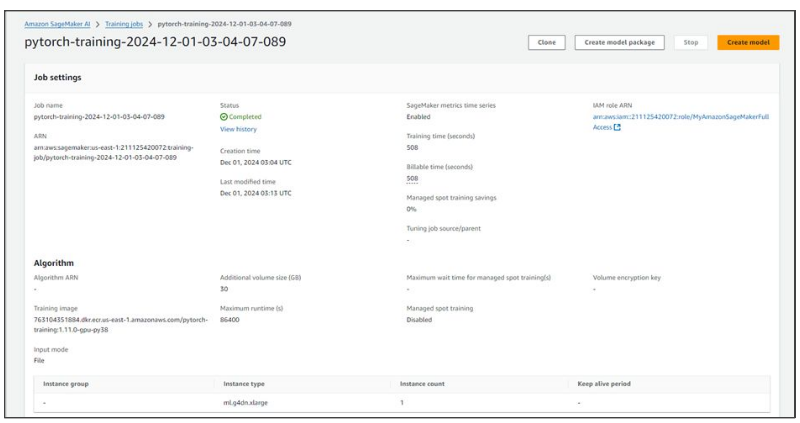
Creating a JupyterLab Instance for Model Training

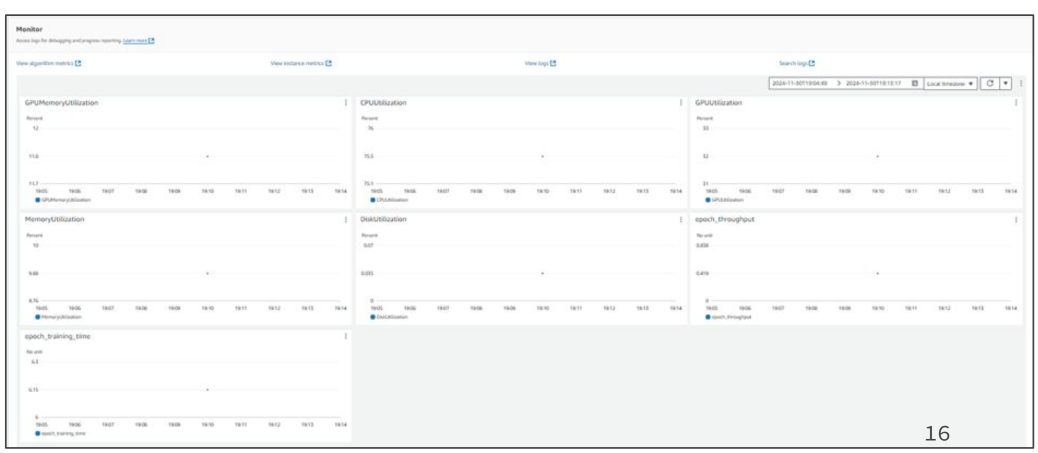
Creating a SageMaker Domain



Launching SageMaker Studio



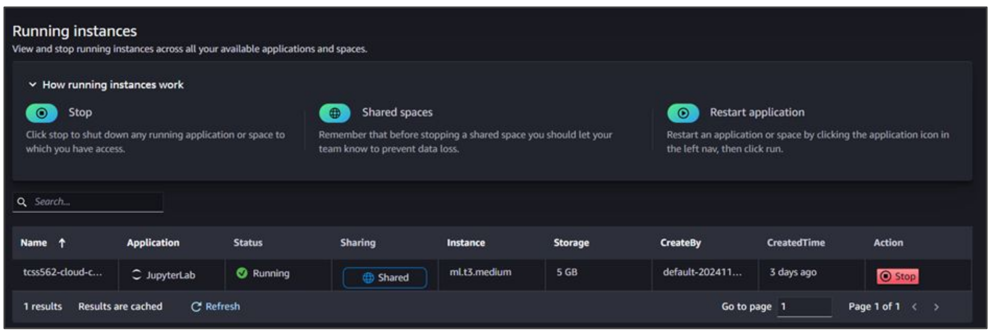


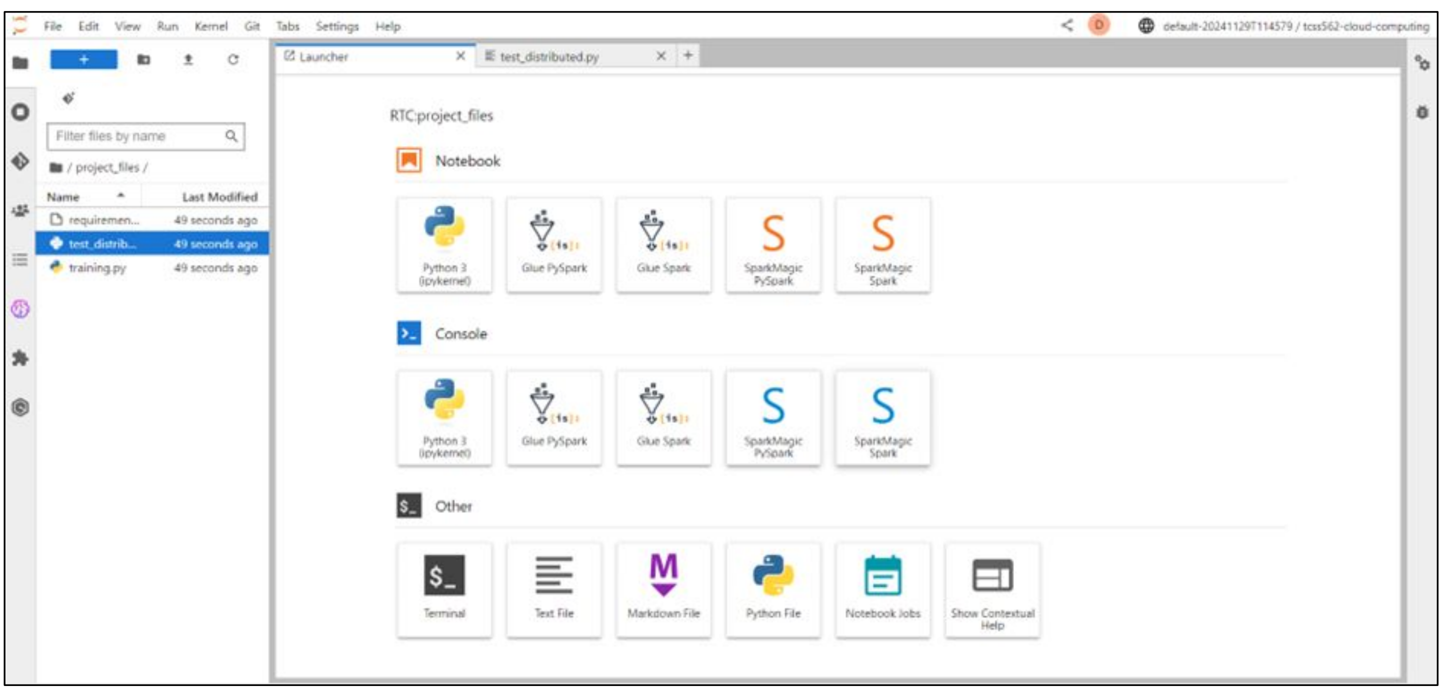


Create JupyterLab Instance

Key Features:

- Shared Workspace
- Comes with pre-configured environment
- Code Editor - development environment
- Jupyter Notebooks





Example Use Case: Distributed Training

Introduction to Distributed Training

Challenge: Training models with Billions of parameters on a single system -

Challenge: Massive datasets (e.g., terabytes of image or text data) cannot fit into the memory of a single machine.

Challenge: Long training cycles on high-end instances can cost tens of thousands of dollars.

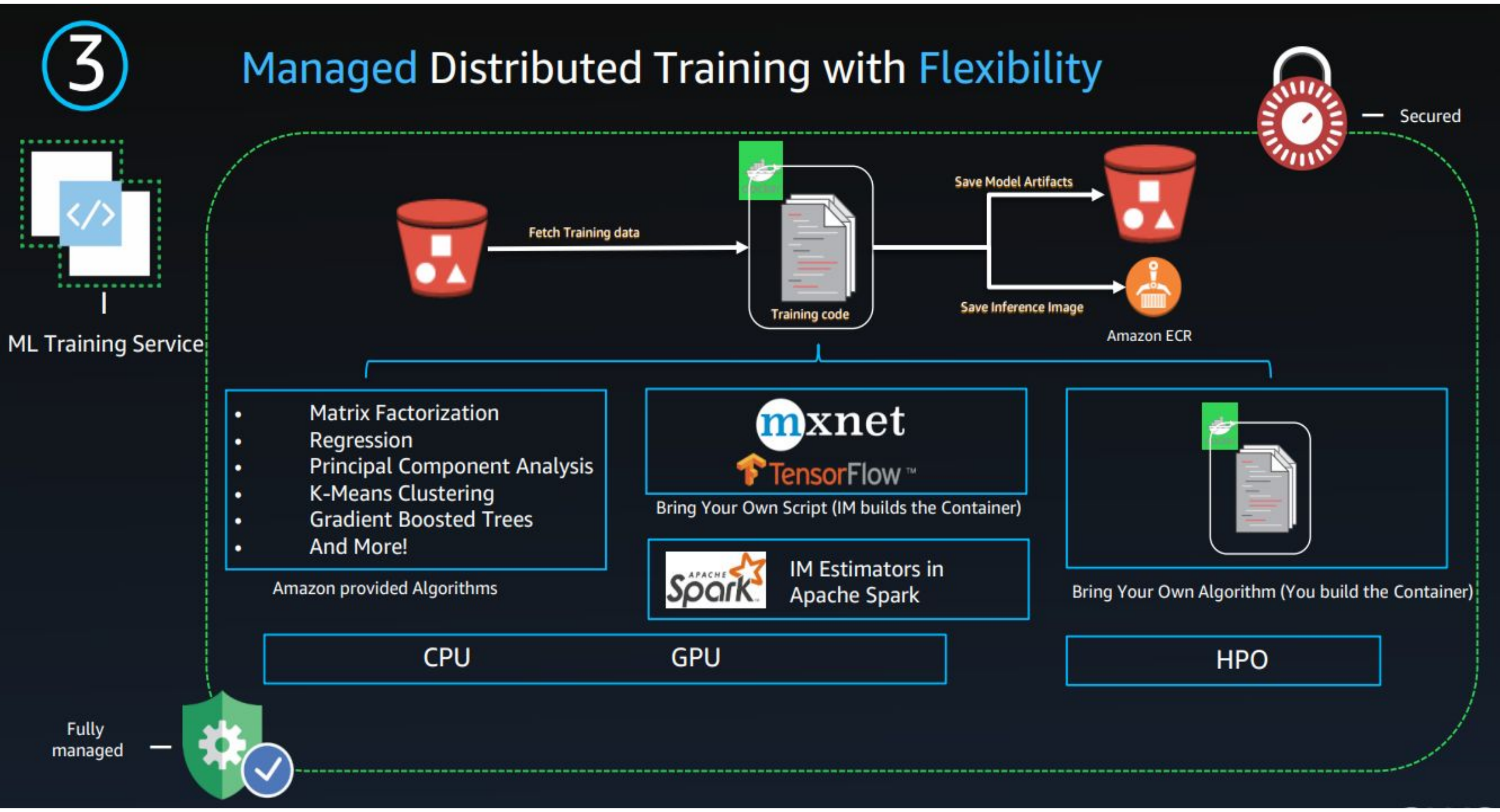
Challenges in Distributed Training (Traditional Setup)

High barriers to entry:

- Setting up and maintaining clusters.
- Managing data distribution and model synchronization.
- Monitoring and debugging distributed workloads.

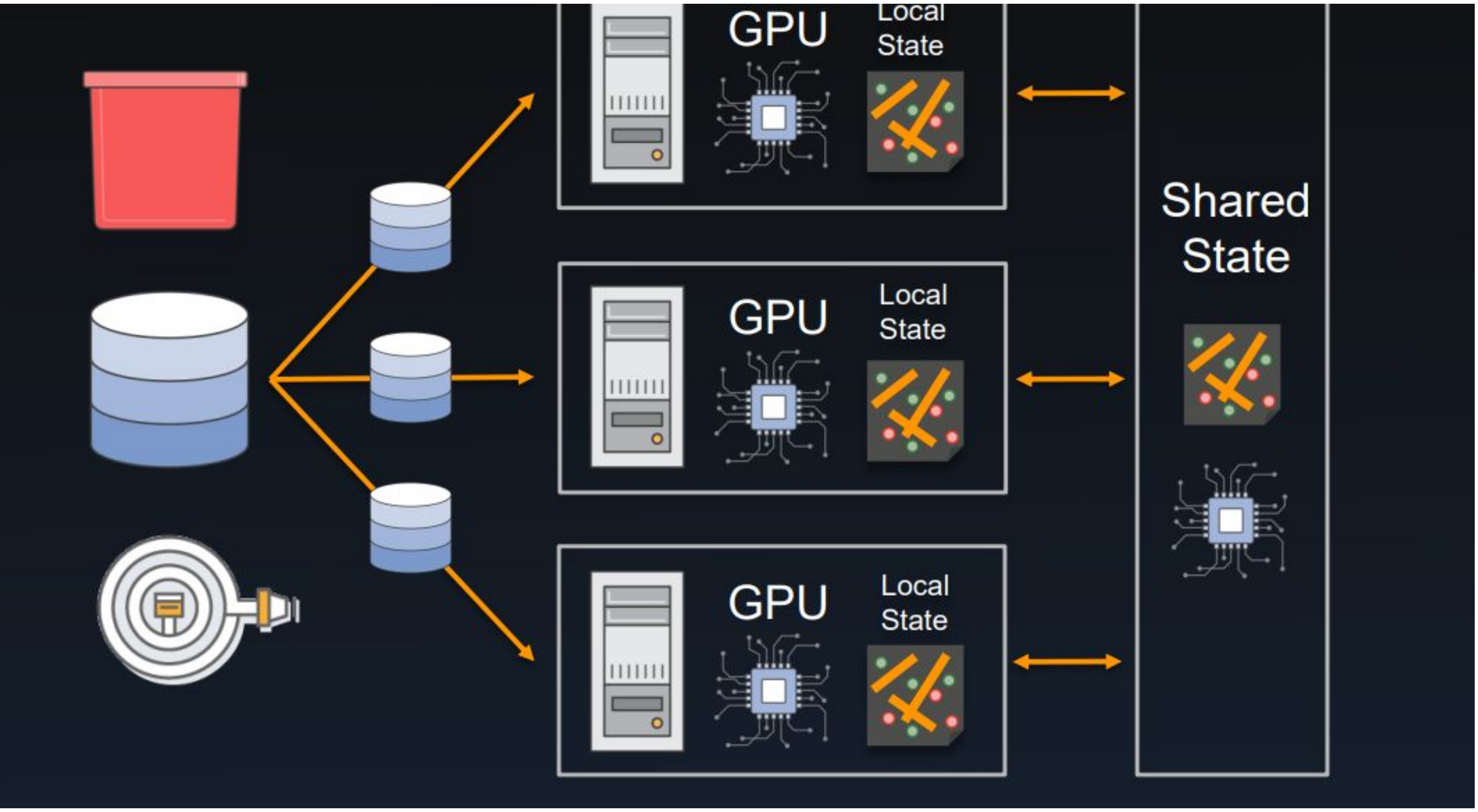
Risk of inefficiencies:

- Uneven resource utilization.
- Long runtimes due to poorly optimized parallelism.

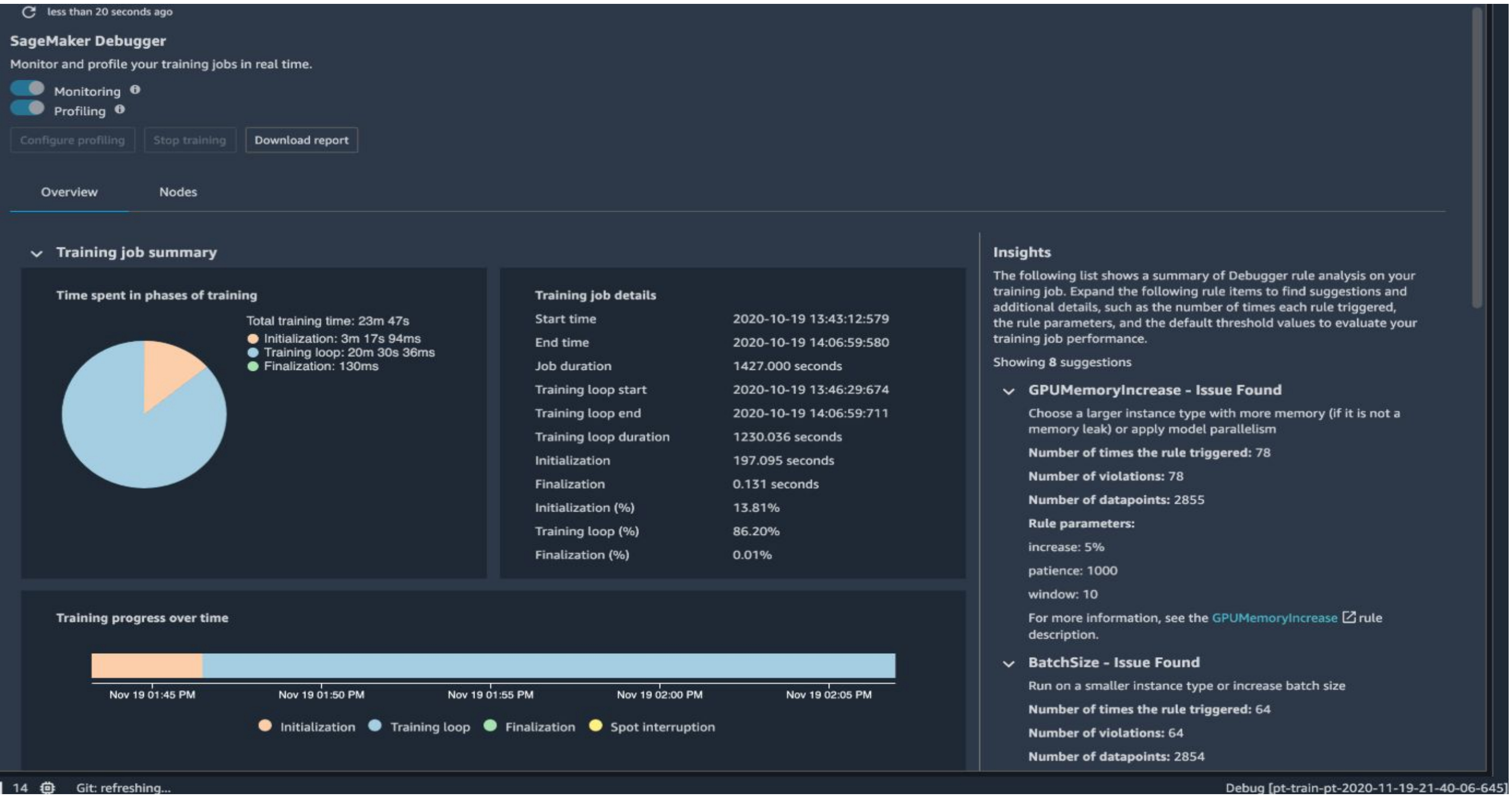


Key Features of SageMaker Distributed Training

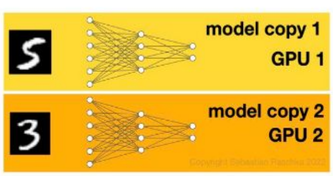
Scalability



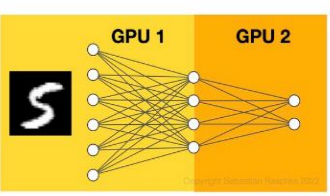
Monitoring and Debugging



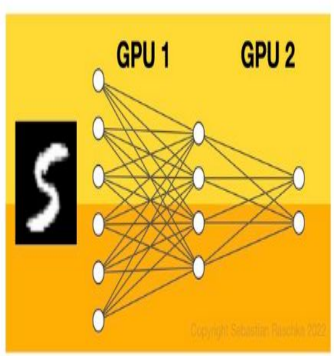
Support for Different kinds of distributed training strategies



Data parallelism



Model parallelism



Tensor parallelism

Access to AWS HPC Clusters enhancements

Real-World Use Cases

The researchers also conducted experiments in which they used SDP to train Mask-RCNN, a neural network with roughly 44 million parameters, on a computer vision task with about 118,000 training examples. The training time was six minutes and 45 seconds on PyTorch and six minutes 12 seconds on TensorFlow, approximately 24% better than the previous record.

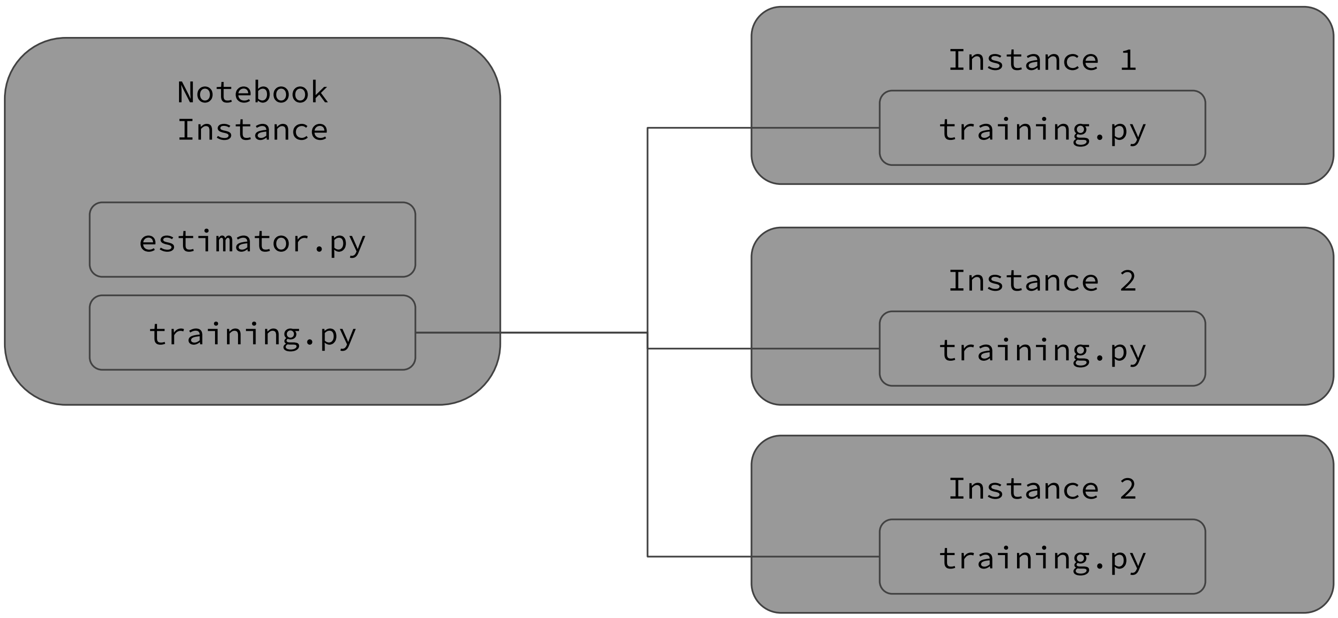
Example Use Case: Distributed Training with PyTorch

What is Distributed Training?

- Splits training tasks across multiple machines and/or GPUs.

SageMaker's Distributed Training Framework:

- Easily configure training clusters.
- Automatic management of resource scaling.



Steps Taken:

- Create a Training Script for the Distributed Job
- Create an Estimator to launch training script across instances
- Set training configurations (i.e. instance-type, preprocessor-worker count, etc.) in estimator.
- Launch from JupyterLab

Conclusion

Criticism of SageMaker

- Prototyping very difficult
  - Very expensive and time consuming to prototype
  - Rebuilds instances from scratch each launch
- Many libraries are proprietary to SageMaker
  - Vendor lock-in
  - Document is sparse for the libraries
  - Difficult dependency management
- High learning curve
  -

Comparison of Costs

Benefits of SageMaker

- Customizable
-

Questions?

Conclusion

Key Takeaways:

- Amazon SageMaker provides an end-to-end managed environment for building, training, and deploying machine learning models.
- SageMaker Studio is an interface for launching instances, viewing training jobs and accessing metrics.